

Lab Record
of
BLOCKCHAIN (IB404)
BACHELOR OF TECHNOLOGY
In
COMPUTER SCIENCE AND ENGINEERING



Session 2022-23

Submitted To: -

Mr. Anand Maurya
IBM
DIT University

Submitted By: -

Name: Shashant
Roll: 190111003
SAP ID: 1000012451
Branch: BTCSE-CSF(P1)

SCHOOL OF COMPUTING
DIT UNIVERSITY, DEHRADUN

(State Private University through State Legislature Act No. 10 of 2013 of Uttarakhand and approved by UGC)
Mussoorie Diversion Road, Dehradun, Uttarakhand - 248009, India.

Aug - Dec.2022

INDEX

S.NO.	NAME OF THE EXPERIMENTS	DATE OF CONDUCTION	SIGNATURE
1	Develop a Code using Media tag.		
2	Develop a Code using Container tag.		
3	Write a program to perform Asynchronous JavaScript.		
4	Write a program to perform JavaScript Promises.		
5	Write steps to create Server in Nodejs		
6	Write steps to deploy a program inside docker.		
7	Setting Up Blockchain Development Environment using IBM Blockchain platform extension		
8	Create Smart Contract and Client App development		
9	Blockchain Applications Guide – Connecting to an existing network		
10	Blockchain Applications Guide – Connecting to an external network		

EXPERIMENT-1

Objective: Develop a Code using Media tag

Program:

```
<!DOCTYPE html>

<html>

  <head>

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <style>

      body {

        background-color: lightgreen;

      }

      @media only screen and (max-width: 600px) {

        body {

          background-color: lightblue;

        }

      }

    </style>

  </head>

  <body>

    <p>This code uses the media tag for html pages to put background color as blue.</p>

  </body>

</html>
```

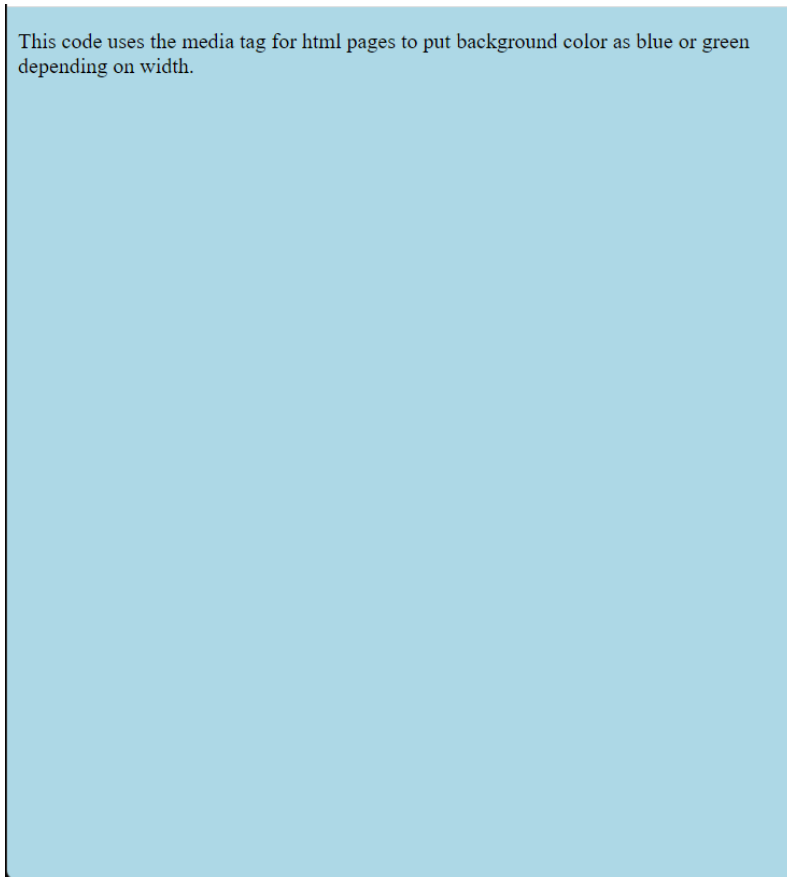


Output:

This code uses the media tag for html pages to put background color as blue or green depending on width.



This code uses the media tag for html pages to put background color as blue or green depending on width.



EXPERIMENT-2

Objective: Develop a Code using Container tag.

Program:

```
<!DOCTYPE html>

<html lang="en">

    <head>

        <meta charset="UTF-8">

        <meta http-equiv="X-UA-Compatible" content="IE=edge">

        <meta name="viewport" content="width=device-width, initial-scale=1.0">

        <title>Experiment 2</title>

    </head>

    <body>

        <h1> This is a container tag for H1 </h1>

        <h2> This is a container tag for H2 </h2>

        <h3> This is a container tag for H3 </h3>

    </body>

</html>
```

Output:

This is a container tag for H1

This is a container tag for H2

This is a container tag for H3

EXPERIMENT-3

Objective: Write a program to perform Asynchronous JavaScript.

Program:

```
<!DOCTYPE html>
<html>
<body>
<h1>Experiment 3</h1>
<h2>JavaScript Asynchronous Function using setInterval() with a Callback</h2>
<p>Using setInterval() to display the time every second (1000 milliseconds).</p>
<h1 id="exp_3"></h1>
<script>
setInterval(myFunction, 1000);
function myFunction() {
  let d = new Date();
  document.getElementById("exp_3").innerHTML=
  d.getHours() + ":" +
  d.getMinutes() + ":" +
  d.getSeconds();
}
</script>
</body>
</html>
```

Output:

Experiment 3

JavaScript Asynchronous Function using setInterval() with a Callback

Using setInterval() to display the time every second (1000 milliseconds).

15:6:14

EXPERIMENT-4

Objective: Write a program to perform JavaScript Promises.

Program:

```
<!DOCTYPE html>
<html>
<body>

<h2>Experiment 4</h2>

<p>JavaScript Promise program to wait 3 seconds (3000 milliseconds) for this page to
change.</p>

<h1 id="exp_4"></h1>

<script>
const myPromise = new Promise(function(myResolve, myReject) {
  setTimeout(function(){ myResolve("Hello World!"); }, 3000);
});

myPromise.then(function(value) {
  document.getElementById("exp_4").innerHTML = value;
});
</script>

</body>
</html>
```

Experiment 4

JavaScript Promise program to wait 3 seconds (3000 milliseconds) for this page to change.

Hello World!

EXPERIMENT-5

Objective: Write steps to create Server in Nodejs

Program:

Steps – Create HTTP Web Server

Following is a step-by-step tutorial, to Create HTTP Web Server in Node.js.

Step 1: Include HTTP Module

Create a .js file with name httpWebServer.js and open in a text editor.

Include the Built-in Node.js module, HTTP, using require function as shown below.

```
var http = require('http');
```

Step 2: Create Server

Create a server using the method createServer() to listen at port numbered 9000.

```
// include http module in the file
var http = require('http');

// create a server
http.createServer(function(req, res) {
  // code to feed or request and prepare response
}).listen(9000); //the server object listens on port 9000
```

Step 3: Prepare response

We shall prepare a response with HTTP header and a message.

```
var http = require('http');

// create a server
http.createServer(function (req, res) {
  // http header
  // 200 - is the OK message
  // to respond with html content, 'Content-Type' should be 'text/html'
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.write('Node.js says hello!'); //write a response to the client
  res.end(); //end the response
}).listen(9000); //the server object listens on port 9000
```


Step 4: Run the Web Server

Run the experiment_5.js file (from previous step) to create and make the server listen at port 9000.

```
$ node experiment_5.js
```

The server will be up and running.

Step 5: Test the Web Server

Open a browser and hit the URL, “http://127.0.0.1:9000/”, to trigger a request to our Web Server.



EXPERIMENT-6

Objective: write steps to deploy a program inside docker.

Program:

Steps:

- 1) Build images.

Step one to make an application in docker is to build the image for the container of the image. To do this, first open the directory of the project. Then create a docker file for the application.

Docker file:

```
# syntax=docker/dockerfile:1
FROM node:16
WORKDIR /usr/src/app
# Install app dependencies
RUN npm install
RUN npm ci --only=production
COPY ..
EXPOSE 8080
CMD ["node", "server.js"]
```

Then using the dockerfile, build an image

```
docker build . -t admin/experiment_6
```

- 2) Run your image as a container.

To run the image as a container, we must use the docker run command.

```
docker run -p 49160:8080 -d admin/experiment_6
```

- 3) Run the container using the container id

To execute the container using the container id, the exec command is to be used.

```
Docker exec -it 41960 /bin/bash
```

```

Creating docker-compose-nodejs-redis_node-app_1 ... done
Creating docker-compose-nodejs-redis_redis-server_1 ... done
Attaching to docker-compose-nodejs-redis_redis-server_1, docker-compose-nodejs-redis_node-app_1
redis-server_1 | 1:C 21 Jul 2019 03:53:08.944 # o000o000o000o Redis is starting o000o000o000o
redis-server_1 | 1:C 21 Jul 2019 03:53:08.944 # Redis version=5.0.5, bits=64, commit=00000000, m
redis-server_1 | 1:C 21 Jul 2019 03:53:08.944 # Warning: no config file specified, using the def
redis-server_1 | 1:M 21 Jul 2019 03:53:08.951 * Running mode=standalone, port=6379.
redis-server_1 | 1:M 21 Jul 2019 03:53:08.951 # WARNING: The TCP backlog setting of 511 cannot b
redis-server_1 | 1:M 21 Jul 2019 03:53:08.952 # Server initialized
redis-server_1 | 1:M 21 Jul 2019 03:53:08.952 # WARNING overcommit memory is set to 0! Backgroun
nd then reboot or run the command 'sysctl vm.overcommit_memory=1' for this to take effect.
redis-server_1 | 1:M 21 Jul 2019 03:53:08.952 # WARNING you have Transparent Huge Pages (THP) su
he command 'echo never > /sys/kernel/mm/transparent_hugepage/enabled' as root, and add it to your
redis-server_1 | 1:M 21 Jul 2019 03:53:08.952 * Ready to accept connections
node-app_1 |
node-app_1 | > @ start /usr/app
node-app_1 | > node index.js
node-app_1 |
node-app_1 | Listening on port 8081

```

EXPERIMENT-7

Objective: Setting Up Blockchain Development Environment using IBM Blockchain platform extension

Program:

Install the extension

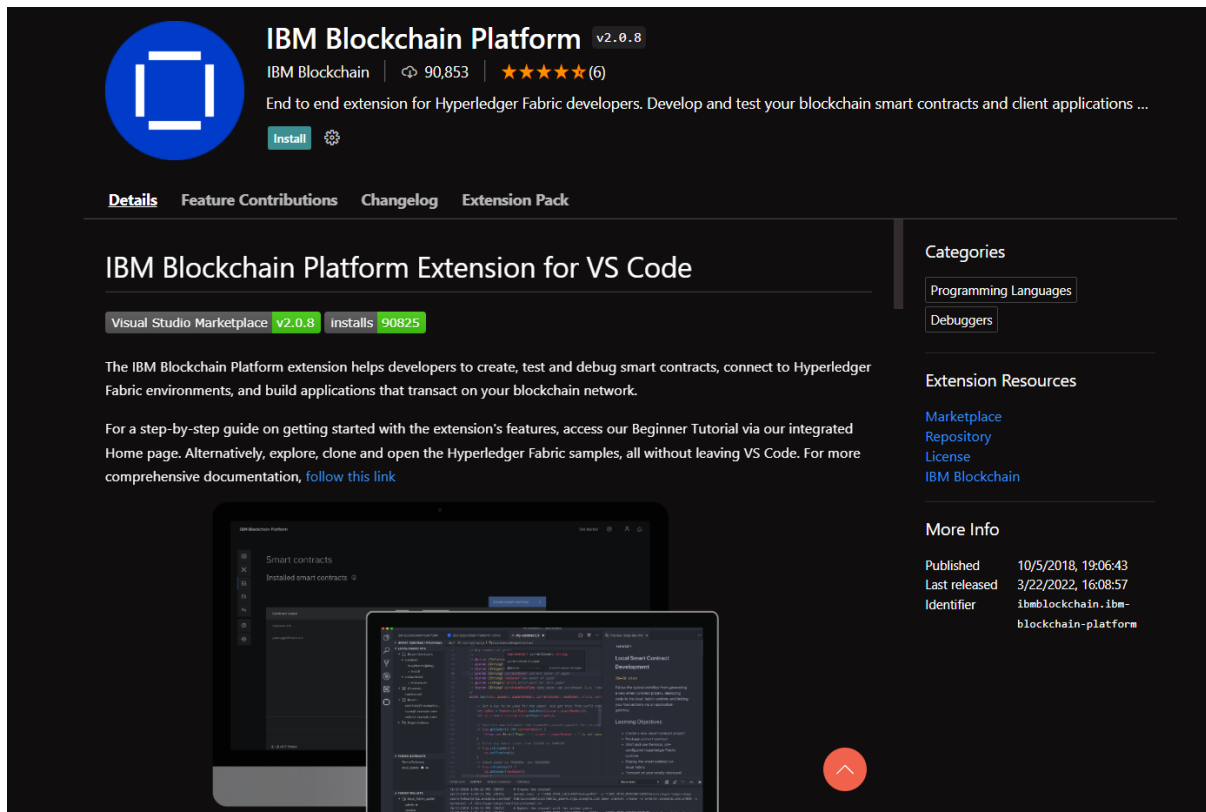
- 1) Ensure that you are running VS Code version 1.40 or greater.
- 2) Go to the Visual Studio Code extension marketplace page or search for IBM Blockchain Platform in the extensions panel within Visual Studio Code.
- 3) Click Install.
- 4) If you are upgrading the extension, you will need to restart VS Code to complete installation of the extension.

After the installation, you can use the IBM Blockchain icon on the left side of VS Code to open the IBM Blockchain Platform panel.¹

Create a smart contract project

You can use the extension to create a new smart contract project in Visual Studio Code. The extension creates a basic smart contract that manages an example asset in the language of your choice. You can use the structure of the example as a starting point for developing your own business logic. The extension provides all the dependencies that are required to deploy your smart contract to an instance of Hyperledger Fabric.

- 1) Click the IBM Blockchain icon to open the IBM Blockchain tab. Click the overflow menu in the smart contracts pane and click Create New Project.
- 2) Select the smart contract type to generate. The Default Contract example is recommended for first-time users and demonstrates how to perform create, read, update, and delete operations to the public ledger that's shared by all network members. The Private Data Contract example demonstrates how to perform create, read, update, delete, and verify operations to a collection, that is private to a single network member.
- 3) Select the language that you want to create a smart contract in. The current options are JavaScript, TypeScript, Go, and Java. Note: If you are deploying the smart contracts to a production network, JavaScript and TypeScript smart contracts require more resources than contracts written in Go.
- 4) Select an asset to be managed by the example contract. For example, bond.
- 5) Create a folder with the name of your project and open it.
- 6) Select how to open your new project. The project folder should now open



IBM Blockchain Platform v2.0.8
 IBM Blockchain | 90,853 | ★★★★★ (6)
 End to end extension for Hyperledger Fabric developers. Develop and test your blockchain smart contracts and client applications ...

Details | Feature Contributions | Changelog | Extension Pack

IBM Blockchain Platform Extension for VS Code

Visual Studio Marketplace **v2.0.8** | installs **90825**

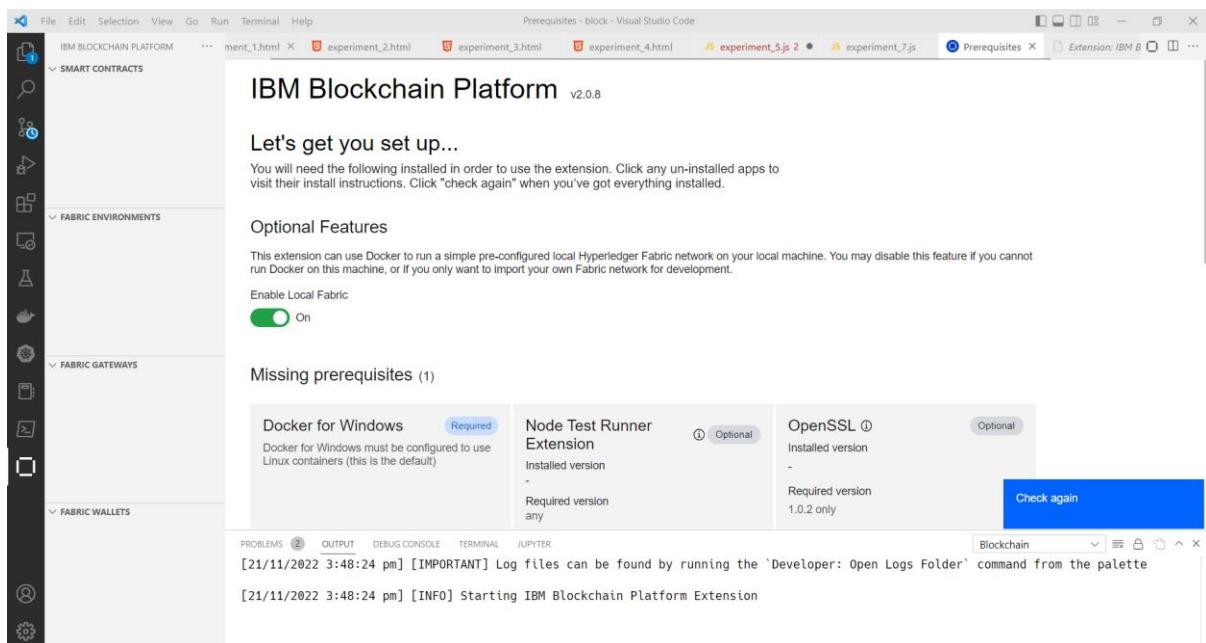
The IBM Blockchain Platform extension helps developers to create, test and debug smart contracts, connect to Hyperledger Fabric environments, and build applications that transact on your blockchain network.

For a step-by-step guide on getting started with the extension's features, access our Beginner Tutorial via our integrated Home page. Alternatively, explore, clone and open the Hyperledger Fabric samples, all without leaving VS Code. For more comprehensive documentation, [follow this link](#)

Categories
 Programming Languages
 Debuggers

Extension Resources
[Marketplace](#)
[Repository](#)
[License](#)
[IBM Blockchain](#)

More Info
 Published: 10/5/2018, 19:06:43
 Last released: 3/22/2022, 16:08:57
 Identifier: ibmblockchain.ibm-blockchain-platform



IBM Blockchain Platform v2.0.8

Let's get you set up...

You will need the following installed in order to use the extension. Click any un-installed apps to visit their install instructions. Click "check again" when you've got everything installed.

Optional Features

This extension can use Docker to run a simple pre-configured local Hyperledger Fabric network on your local machine. You may disable this feature if you cannot run Docker on this machine, or if you only want to import your own Fabric network for development.

Enable Local Fabric: ☒ On

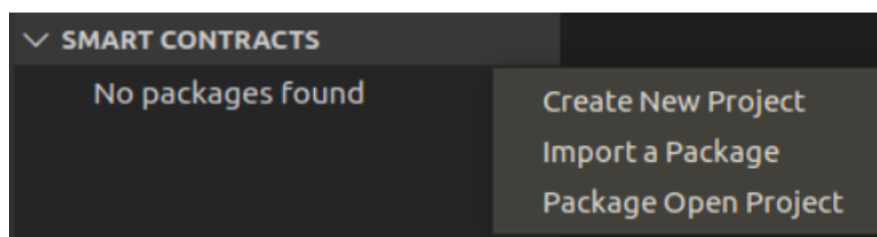
Missing prerequisites (1)

Prerequisite	Status	Version	Action
Docker for Windows	Required	-	Check again
Node Test Runner Extension	Optional	Installed version: -, Required version: any	Check again
OpenSSL	Optional	Installed version: -, Required version: 1.0.2 only	Check again

PROBLEMS [2] | **OUTPUT** | **DEBUG CONSOLE** | **TERMINAL** | **JUPYTER**

[21/11/2022 3:48:24 pm] [IMPORTANT] Log files can be found by running the 'Developer: Open Logs Folder' command from the palette

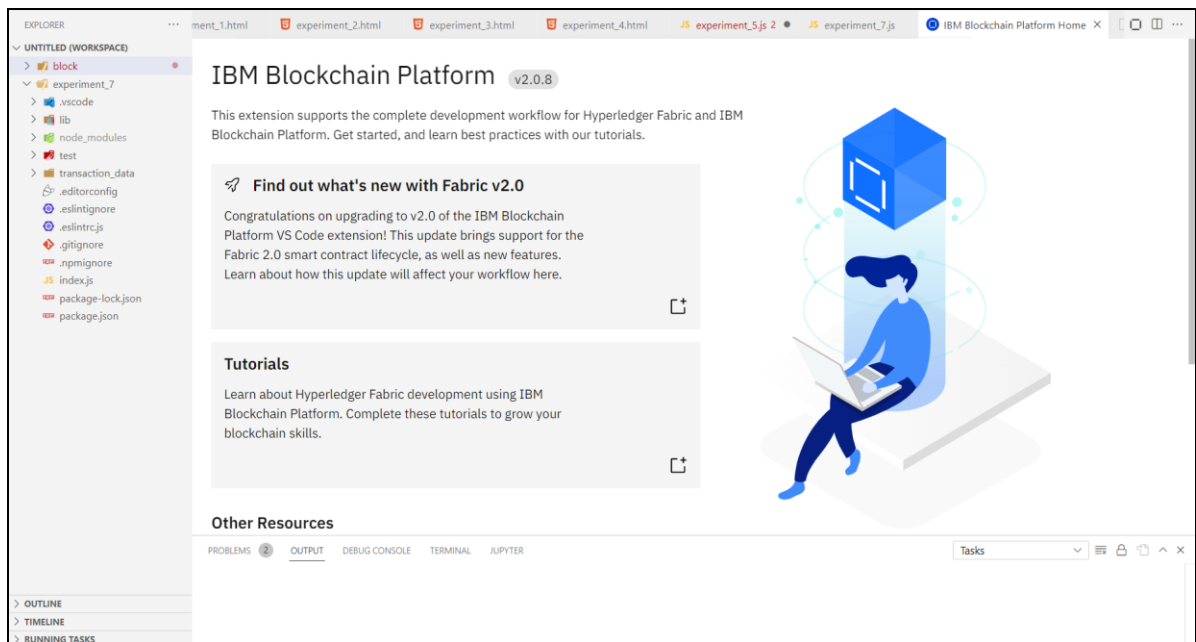
[21/11/2022 3:48:24 pm] [INFO] Starting IBM Blockchain Platform Extension



SMART CONTRACTS

No packages found

Create New Project
 Import a Package
 Package Open Project



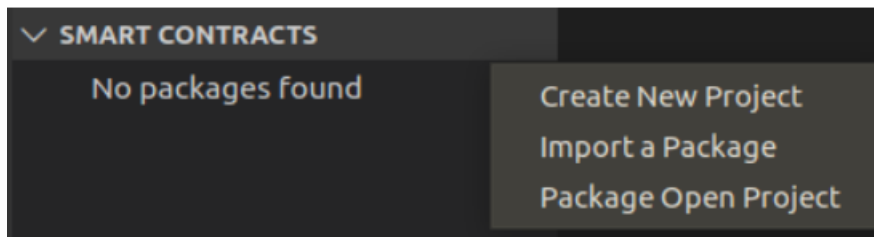
EXPERIMENT-8

Objective: Create Smart Contract and Client App development

Program:

Step 1) Create a smart contract project

You can use the extension to create a new smart contract project in Visual Studio Code. The extension creates a basic smart contract that manages an example asset in the language of your choice.



Step 2) After the project is loaded, write the smart contract

Once, the project is loaded, the index.js file can be used to manipulate and create smartcontracts. Code for index.js:

```
/*
 * SPDX-License-Identifier: Apache-2.0
 */

'use strict';

const MyAssetContract = require('./lib/my-asset-contract');

module.exports.MyAssetContract = MyAssetContract;
module.exports.contracts = [ MyAssetContract ];
```

Here, my-asset-contract is the smartcontract which is called onto the client application.

```
/*
 * SPDX-License-Identifier: Apache-2.0
 */

'use strict';

const { Contract } = require('fabric-contract-api');

class MyAssetContract extends Contract {
```

```
async createMyAsset(ctx, myAssetId, value) {  
  const exists = await this.myAssetExists(ctx, myAssetId);  
  if (exists) {  
    throw new Error(`The my asset ${myAssetId} already exists`);  
  }  
  const asset = { value };  
  const buffer = Buffer.from(JSON.stringify(asset));  
  await ctx.stub.putState(myAssetId, buffer);  
}
```

```
module.exports = MyAssetContract;
```

Hence, the following code will produce the smartcontract and the client app.

EXPERIMENT-9

Objective: Blockchain Applications Guide – Connecting to an existing network

To connect to an existing network, we must first join a network by using the IBM blockchain platform console.

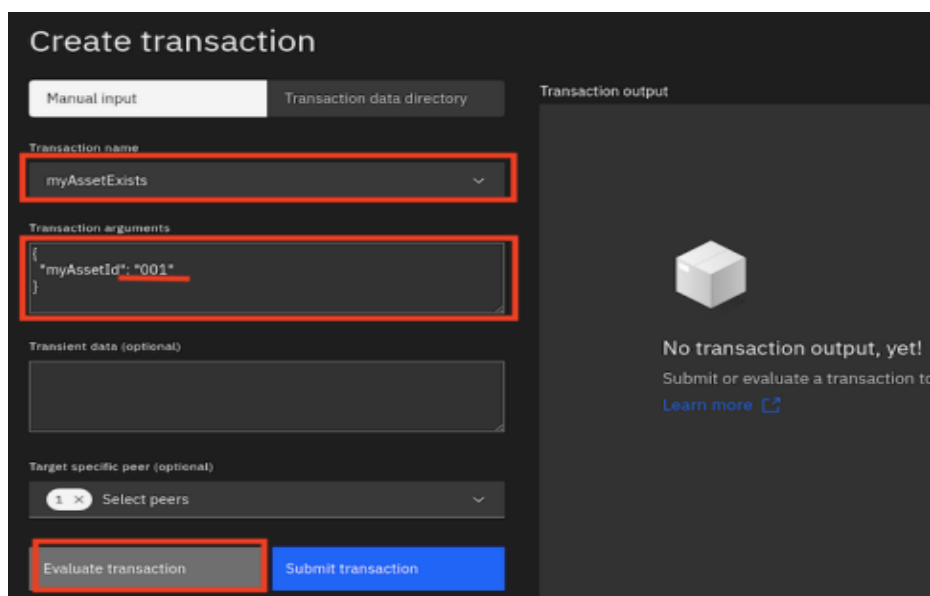
In order to submit transactions in Hyperledger Fabric you will need an identity, a wallet, and a gateway.

Identities, wallets and gateways the resources that you can access in a Hyperledger Fabric network are determined according to your identity; that is why it's called a permissioned blockchain. Your identity is typically represented by an X.509 certificate issued by your organization, and stored in your wallet.

Once you have an identity and a wallet, you can create a gateway that allows you to submit transactions to a network. A gateway represents a connection to a set of Hyperledger Fabric networks. If you want to submit a transaction, whether using VS Code or your own application, a gateway makes it easy to interact with a network: you configure a gateway, connect to it with an identity from your wallet, choose a particular network, and start submitting transactions using a smart contract that has been deployed in that network.

A gateway is configured using a connection profile, which identifies a single peer in the network as an initial connection point. We are going to use a pre-configured gateway that was created when we started the one organization network.

Output:



```
DEBUG CONSOLE  PROBLEMS  OUTPUT  TERMINAL
[4/24/2020 1:03:32 PM] [INFO] - src/transaction_data/my-asset-transactions.txdata
[4/24/2020 1:03:52 PM] [INFO] installSmartContract
[4/24/2020 1:03:52 PM] [SUCCESS] Successfully installed on peer Org1Peer1
[4/24/2020 1:07:09 PM] [SUCCESS] Successfully instantiated smart contract
[4/24/2020 1:14:43 PM] [INFO] connect
[4/24/2020 1:14:56 PM] [INFO] connect
[4/24/2020 1:15:37 PM] [SUCCESS] Connecting to 1 Org Local Fabric - Org1
[4/24/2020 1:17:49 PM] [INFO] evaluateTransaction
[4/24/2020 1:19:36 PM] [INFO] evaluating transaction myAssetExists with args
[4/24/2020 1:19:36 PM] [SUCCESS] Returned value from myAssetExists: false
Successfully evaluated transaction
```

Transaction name

myAssetExists

Transaction arguments

```
{
  "myAssetId": "001"
}
```

The returned value seen in the Transaction output is now 'true'.

```
Transaction output
Returned value from myAssetExists: true
```

A4.14: Submit the "updateMyAsset" transaction to change the value of the "001" key to "The Hay Wain". The JSON Transaction arguments will be:

```
{
  "myAssetId": "001",
  "value": "The Hay Wain"
}
```

No value will be returned from the transaction.

A4.15: Evaluate the "readMyAsset" transaction to return the updated asset. Specify "001" for myAssetId.

```
Transaction output
Returned value from readMyAsset: {"value":"The Hay Wain"}
```

A4.16: Finally, submit the "deleteMyAsset" transaction to delete the "001" asset from the world state.

```
DEBUG CONSOLE  PROBLEMS  OUTPUT  TERMINAL
[4/24/2020 1:28:23 PM] [INFO] submitTransaction
[4/24/2020 1:28:28 PM] [INFO] submitting transaction deleteMyAsset with args 001 on channel mychannel
[4/24/2020 1:28:30 PM] [SUCCESS] No value returned from deleteMyAsset
```

Note carefully this last transaction! We have added a delete transaction to the blockchain, which has resulted in an empty state database for key "001". It is perfectly possible to delete assets from the world state, but submitted transactions are always added to the ledger. The blockchain records the changes that have happened to the world state database, which can include deleting records as well as adding and modifying them.

EXPERIMENT-10

Objective: Blockchain Applications Guide – Connecting to an external network

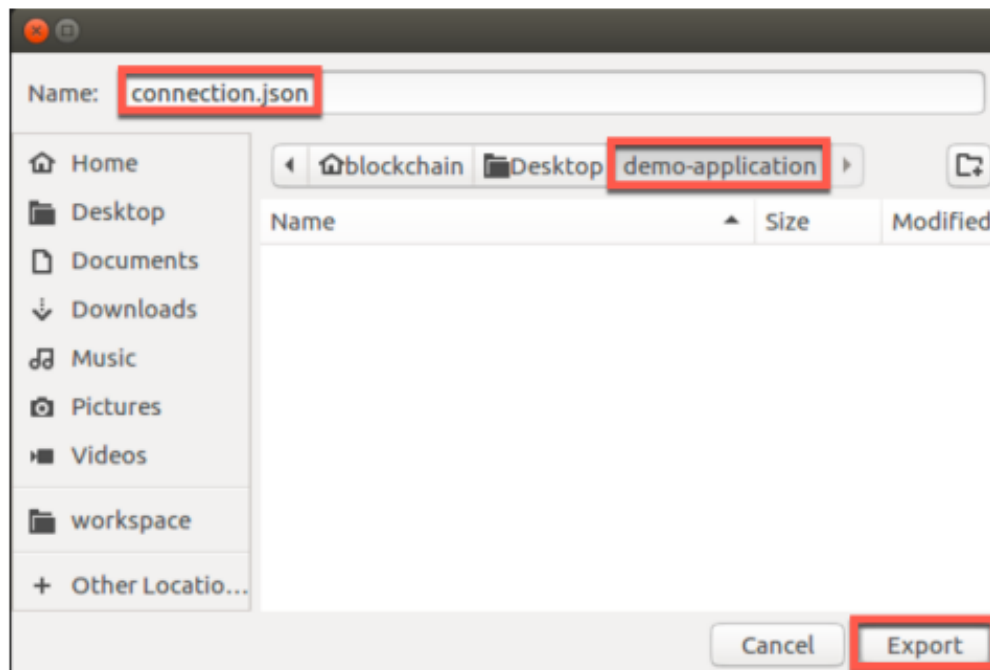
Export the network details

As we have seen, to interact with a Hyperledger Fabric network it is necessary to have:

- a connection profiles
- a wallet containing one or more identities

Our sample application will use the same identity and connection profile used by VS Code to interact with the sample network.

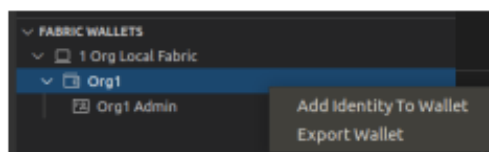
We will start by exporting a connection profile.



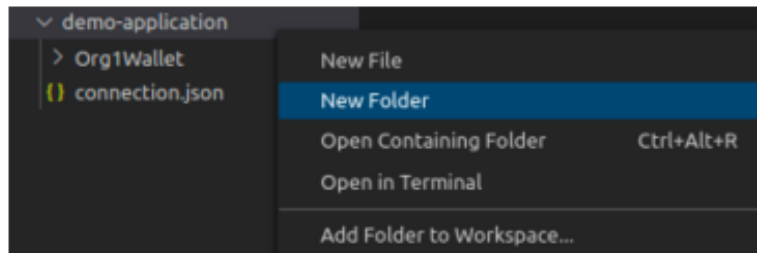
We will now export our wallet.

A5.4: In the Fabric Wallets view, expand '1 Org Local Fabric', right click 'Org1' and select 'Export Wallet'.

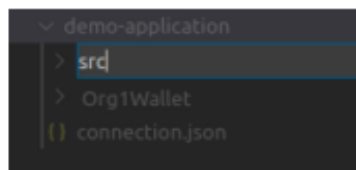
Take care not to click on the Orderer organization's wallet by mistake.



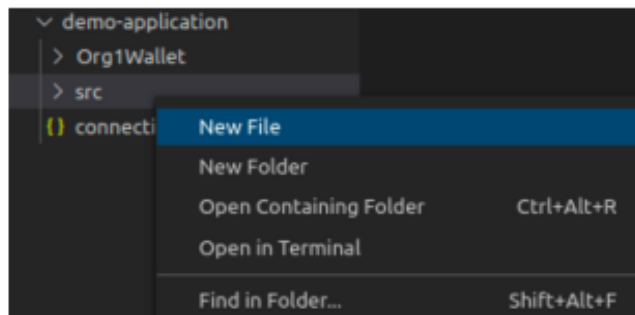
A5.5: Navigate into the 'demo-application' folder, change the name to 'Org1Wallet' and click Export to save the wallet.



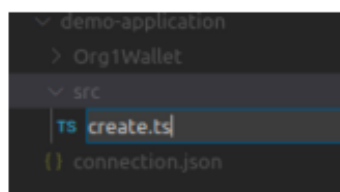
☐ A5.10: Name the folder 'src'.



☐ A5.11: Right-click 'src' and select 'New File'.



☐ A5.12: Name the file 'create.ts'.



☐ A5.13: In the editor view for the new create.ts file, copy and paste the following text. (The contents are also available [here](#)).

```
import { Gateway, Wallets } from 'fabric-network';
import * as path from 'path';
import * as fs from 'fs';

async function main() {
  try {
    // Create a new file system based wallet for managing identities.
    const walletPath = path.join(process.cwd(), 'Org1Wallet');
```

```
const wallet = await Wallets.newFileSystemWallet(walletPath);
console.log(`Wallet path: ${walletPath}`);

// Create a new gateway for connecting to our peer node.
const gateway = new Gateway();
const connectionProfilePath = path.resolve(__dirname, '..',
'connection.json');
const connectionProfile =
JSON.parse(fs.readFileSync(connectionProfilePath, 'utf8')); // eslint-
disable-line @typescript-eslint/no-unsafe-assignment
const connectionOptions = { wallet, identity: 'Org1 Admin', discovery:
{ enabled: true, asLocalhost: true } };
await gateway.connect(connectionProfile, connectionOptions);

// Get the network (channel) our contract is deployed to.
const network = await gateway.getNetwork('mychannel');


// Get the contract from the network.
const contract = network.getContract('demo-contract');

// Submit the specified transaction.
await contract.submitTransaction('createMyAsset', '002', 'Night
Watch');
console.log('Transaction has been submitted');

// Disconnect from the gateway.
gateway.disconnect();

} catch (error) {
console.error('Failed to submit transaction:', error);
process.exit(1);
}
}
void main();
```

Your file should be 38 lines long. We will look through what the application is doing later on in this tutorial.

 A5.14: Save the file ('File' -> 'Save').

Saving the file will change the tab for the editor to show a cross; a solid circle here means that you have unsaved changes:



When you save, you will see various errors reported by VS Code. This is because we have not yet configured the set of external dependencies.